Learning Efficient Interpretable Policies on Experimental Data

Han Wu^, Sarah Tan*, Weiwei Li*, Mia Garrard*, Hanson Wang*,
Daniel Jiang*, Adam Obeng*, Eytan Bakshy*

^Stanford University; *Facebook
Corresponding emails: hanwu71@stanford.edu; sarahtan@fb.com

1 Introduction

Unlike conventional A/B testing where users are randomly assigned to receive a treatment [4, 11], internet companies are increasingly using machine learning models for personalized experimentation to create personalized policies [2]. Such policies assign, for each user, the best predicted treatment for that user.

A popular approach for personalized experimentation is to train heterogeneous treatment effect (HTE) models [7, 9], and then assign the treatment group that led to the highest predicted treatment effect for that user. However, there are several challenges with this approach. Firstly, when there are multiple (not just binary) outcomes, the creation of a policy is no longer as trivial as assigning the treatment group that maximizes a single treatment effect. Secondly, as dataset size and the number of features used for HTE modeling increases, the cost of maintaining such models in a production environment increases. Finally, the black-box nature of popular HTE models, that may consist of uninterpretable base learners such as gradient boosted decision trees, but are also combined in different ways to generate the treatment effect prediction, may deter uptake of such models, especially for critical applications.

Here we focus on learning interpretable policies based on if-else rules of the user features. Figure 1 provides an example of an interpretable policy with three treatment groups, using only two features – age and gender. Such a policy is easy to implement in a production environment and avoids the need to maintain a HTE model.

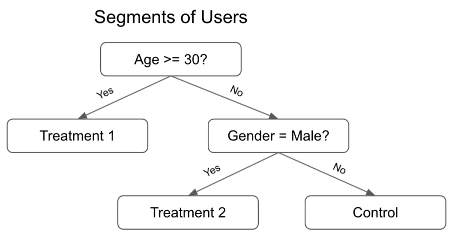

Figure 1: **Rule-based policy**

There are different ways to generate interpretable policies. We briefly summarize some of these approaches and highlight how the methods we propose present a contribution. Existing works [1, 3] construct tree policies based on counterfactual estimation. However they both focus on constructing exactly optimal trees, which is prohibitively slow in a large scale. We consider methods that involve distilling black-box heterogeneous treatment effect models [5, 6, 8] and methods that modify interpretable models such as decision trees to find segments with elevated treatment effects [1, 3]. We also propose two new approaches to ensemble multiple interpretable policies while still remaining interpretable. Inline with all the approaches, we propose work on the general setting of more than two treatment groups and more than a single outcome, which is needed for real-world settings.

2 Methodology

We introduce two types of methods to generate interpretable policies. The first type is what we will call *non-model based methods* that do not require training a treatment effect model. The second type is *distillation methods*, based on distilling treatment effect models. We also consider two heuristic methods for ensembling the individual policies.

2.1 Non-Model Based Methods

With non-model based methods, the goal is to find a segment that has large sample average treatment effects. Suppose we have the treatment indicator $W_i \in \{0, 1, \ldots, K\}$ where $W_i = 0$ indicates control group and we denote the number of users in each group as $N_j, j = 0, 1, \ldots, K$. We define the metric of a sample $\mathcal{D}$ as:

$$M(\mathcal{D}) = \max_{j=0,1,\ldots,K} \left(\frac{1}{N_j} \sum_{W_i=j} Y_i - \frac{1}{N_0} \sum_{W_i=0} Y_i \right)$$

Our method builds a tree and considers binary splits that improve this metric. **In our greedy implementation**, we only consider a split if both the left and right child have an improvement over the parent node. The resulting segments will be given the treatment that achieves the maximum in the metric definition. **In our iterative implementation**, we run several iterations of tree splitting. For each iteration we keep the best segment and exclude it in the next run. We consider a split if one of the children improves the value of metric compared to the parent node.

2.2 Distillation Methods

Our distillation methods build on predictions from treatment effect model. Suppose we have a training dataset of the form $(X_i, \hat{Y}_i(0), ..., \hat{Y}_i(K))$ where $\hat{Y}_i(0), ..., \hat{Y}_i(K)$ are the predictions of potential outcomes/treatment effects for each treatment condition. To obtain an interpretable policy we consider two approaches.

Distilling the predictions: In this approach, we directly solve the following optimization problem in the space of trees with pre-determined maximum depth

$$\Pi^* = \text{argmax}_{\Pi} \frac{1}{n} \sum_{i=1}^{n} \hat{Y}_i(\Pi(X_i)).$$

To achieve a scalable implementation, we solve the optimization problem greedily instead of resorting to exact tree search over the whole space of possible trees.

Distilling the policy: In this approach we first use the predictions to get a policy on the training dataset, i.e. $\Pi(X_i) = \text{argmax}_W \hat{Y}_i(W)$. Then we train a tree classifier to solve the classification problem on the dataset $(X_i, \Pi(X_i))_{i=1}^n$.

2.3 Ensemble Heuristics

We could use the above methods to generate several interpretable policies. However, different policies may have different strengths in different regions, and simply training the above methods with deeper depth does not necessarily improve the resulting policy in our experiments. We leveraged ensemble learning to identify such regions, with the hope of generating a better policy. In other words, suppose we have access to M policies $\Pi_1, \Pi_2, \ldots, \Pi_M$, we want to train a policy $\widetilde{\Pi} : \mathcal{X} \to \{1, 2, \ldots, M\}$ and the final policy to deploy would be $\Pi_{\widetilde{\Pi}(X_i)}(X_i)$. We introduce two ways of doing so.

Ensemble based on Uniform Exploration: This method is inspired by the explore-then-exploit paradigm in the contextual bandit literature [10]. However, since this method is performed offline, we use predictions from treatment effect model when the observed outcome is not revealed in the dataset. Specifically, we uniformly assign one of the M candidate policies to the users in the validation dataset. The selected candidate policy would tell us an action $a \in \{0, 1, \ldots, K\}$. In the online setting, we would observe an outcome $Y = Y(a)$. In this offline setting, our data is of the form $(X_i, W_i, Y_i = Y_i(W_i))$. If $W_i = a$ we use Y_i as the observed outcome O_i; if $W_i \neq a$, we use the prediction $\hat{Y}_i(a)$ as the observed outcome O_i. We then have access to a new dataset $(X_i, A_i \in \{1, 2, \ldots, M\}, O_i)$ and could train a tree policy of any depth using IPS to impute the partial feedback and a cost sensitive classification oracle (e.g. the one we implemented in 2.2).

Ensemble using Offline Evaluation: Due to the computational burden of this method, particularly when there are many policies to ensemble, we only create one additional split. For each feature and splitting point, we consider all possible assignments of the candidate policies to the left and right child. We then evaluate the resulting policy offline on the validation dataset using inverse propensity scoring (IPS). The resulting ensemble is defined by the best split.

3 Results

Our preliminary results are that interpretable policies are comparable to black-box policies in some cases. The ensemble methods tend to improve over candidate policies used in ensembling. Interestingly, when no heterogeneity is present, interpretable policies are able to pick the best individual policy if no heterogeneity. Despite their simplicity, non-model based methods are able to achieve the same level of performance as other methods that require training treatment effect models.

None of the interpretable policies used more than three features, which makes them easy to maintain. Planned future work is to evaluate the approaches on more datasets.

References

[1] Maxime Amram, Jack Dunn, and Ying Daisy Zhuo. Optimal policy trees. *arXiv preprint arXiv:2012.02279*, 2020.

[2] Pavlos Athanasios Apostolopoulos, Zehui Wang, Hanson Wang, Chad Zhou, Kittipat Virochsiri, Norm Zhou, and Igor L. Markov. Personalization for web-based services using offline reinforcement learning. *arXiv:2102.05612*, 2021.

[3] Susan Athey and Stefan Wager. Policy learning with observational data. *Econometrica*, 89(1):133–161, 2021.

[4] Eytan Bakshy, Dean Eckles, and Michael S Bernstein. Designing and deploying online field experiments. In *Proc. of WWW' 14*, pages 283–292, 2014.

[5] Max Biggs, Wei Sun, and Markus Ettl. Model distillation for revenue optimization: Interpretable personalized pricing. In *International Conference on Machine Learning*, pages 946–956. PMLR, 2021.

[6] Wojciech M Czarnecki, Razvan Pascanu, Simon Osindero, Siddhant Jayakumar, Grzegorz Swirszcz, and Max Jaderberg. Distilling policy distillation. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1331–1340. PMLR, 2019.

[7] Sören R Künzel, Jasjeet S Sekhon, Peter J Bickel, and Bin Yu. Metalearners for estimating heterogeneous treatment effects using machine learning. *Proceedings of the national academy of sciences*, 116(10):4156–4165, 2019.

[8] Maggie Makar, Adith Swaminathan, and Emre Kıcıman. A distillation approach to data efficient individual treatment effect estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4544–4551, 2019.

[9] Xinkun Nie and Stefan Wager. Quasi-oracle estimation of heterogeneous treatment effects. *Biometrika*, 108(2):299–319, 2021.

[10] Aleksandrs Slivkins. Introduction to multi-armed bandits, 2021.

[11] Ya Xu, Nanyu Chen, Adrian Fernandez, Omar Sinno, and Anmol Bhasin. From infrastructure to culture: A/b testing challenges in large scale social networks. In *Proc. KDD*, pages 2227–2236, 2015.